Locating Geometric Primitives by Pruning the Parameter Space

Clark F. Olson* and Larry H. Matthies
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive, MS 107-102
Pasadena, CA 91109

Abstract

This paper examines the extraction of geometric primitives from two- and three-dimensional image data. The geometric primitives are represented by parametric manifolds in the image space, such as circles, planes, and cylinders. An explicit error model allows the primitives to be extracted robustly. The primitives are located by exploring the parameter space, which is structured as a hierarchy of cells. This results in a search strategy that is robust to distractors, missing data, and noise and that does not require an initial estimate of the positions of the geometric primitives. We examine, in particular, the use of these techniques to detect craters on planetary bodies by extracting circles in two-dimensional edge data and to find unexploded ordnance in test ranges by locating cylinders in three-dimensional range data and real examples are shown.

1 Introduction

The extraction of geometric primitives from image data is a useful tool in many applications. Two examples that will be examined in this work are detecting craters on planetary bodies by extracting circles from edge data, and locating unexploded ordnance (UXO), which requires the extraction of cylinders from stereo range data.

Several methods have been used to extract geometric primitives from image data. The most popular (the Hough transform and its variations (see [9, 10] for reviews of such techniques). These methods map the image features into the space of possible primitive parameters and then search for peaks in this parameter space, since these peaks correspond to likely geometric primitives.

Other approaches include the use of robust statistics to fit the image data in the presence of noise and outliers [7, 13], methods that hypothesize instances of positions using small sets of data features and then test each position that is hypothesized [3, 14], minimization of a cost function through iterative optimization [15], and region growing [2].

A key drawback to many of these methods is that it is difficult to both propagate the effects of localization error in the data features and handle large amounts of distracting data (a particular primitive may consist of a small fraction of the total data). The methods of Chen [7] and Besl and Jain [2] can deal with these problems, but both methods require an initial estimate of the primitive position and then iteratively refine the fit.

In this work, we take an approach that deals with the effects of localization error explicitly through the use of a bounded error model, can handle large amounts of distracting data, and does not require an initial estimate of the primitive positions. Our approach is inspired by research on object recognition in which the parameter space is recursively divided and pruned [4, 8]. We retain the splitting and pruning search strategy, but rather than using discrete points as our model, we use a parameterized manifold to represent the geometric primitives. Similar ideas are also used in the Fast Hough Transform [11], although this method does not propagate localization error.

We note that it is possible to use object recognition techniques directly in the extraction of geometric primitives, by constructing a canonical primitive composed of a set of discrete points, but there are several disadvantages to this approach. First, if we do not know the scale of the primitive, the sampling of points on the primitive will either be too coarse, and thus a poor representation, or too fine, and very time consuming. Second, some primitives, such as cylinders, have potentially infinite extent. If we place an artificial bound on the extent of such primitives, it can lead to difficulties in the extraction, and it adds another degree of freedom upon which we must search. Such

*http://robotics.jpl.nasa.gov/people/olson/homepage.html

as translations along the axis of a cylinder.

In this work, geometric primitives are extracted from the image data by searching for parameters corresponding to primitives that satisfy an acceptance criterion based on how many of the data features are fit by the primitive up to a bounded error. In order to perform this search, we consider rectilinear cells of the parameter space. For each such cell that is examined, we determine whether the cell could contain the parameters for a primitive that meets the acceptance criterion. If not, then the cell is pruned, otherwise the cell is split into two subcells and the subcells are examined recursively. When a very small cell is reached, we test the primitive at the center of the cell to determine if it meets the acceptance criterion.

Several techniques are used to improve the speed of the search. A hierarchy of the image features is constructed that allows the pruning of multiple features at a time, so that not every data feature must be considered explicitly for each cell of the transformation space. Robust random sampling techniques also used to improve the speed of the search.

The balance of this paper explores these ideas in detail. Section 2 discusses the method by which geometric primitives are represented. Section 3 describes the basic method for searching the parameter space. The optimizations that are used to speed up the search are discussed in Section 4. Section 5 gives examples of the results that have been achieved with real images. Finally, Section 6 summarizes the contributions of this paper.

2 Parameterization of primitives

A class of primitives can be represented as parametric manifolds described by the equation $f(X;P) = 0$, where X is a vector of the data feature parameters and P is a vector of the unknown parameters. We will primarily consider two cases in this paper, the extraction of circles from two-dimensional edge data and the extraction of cylinders from three-dimensional range data.

2.1 Circles

In the case where we want to extract circles from a two-dimensional edge image, our data features have two parameters, the x and y position of the edge point, so in this case, $X = [x\ y]^t$. Circles have three degrees of freedom, parameterized by, for example, the coordinates of the center, (x_c, y_c), and the radius, r, so we have $P = [x_c\ y_c\ r]^t$. Our parameterizing equation is:

$$(x - x_c)^2 + (y - y_c)^2 - r^2 = 0$$

If we have information on the local orientation of the pixels (for example the normal to the edge chain), we can use $X = [x\ y\ \theta]$ and add the following constraint:

$$\tan\theta = \frac{y - y_c}{x - x_c}$$

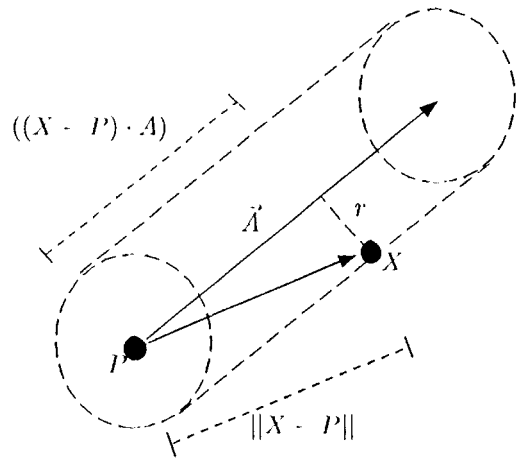


Figure 1: Parameterization of a cylinder.

2.2 Cylinders

When extracting cylinders from range data, our data features have three parameters, $X = [x\ y\ z]^t$. Cylinders have five degrees of freedom. We can parameterize cylinders by: A, a unit vector in the direction of the axis (two degrees of freedom), P, the point on this axis closest to the origin (two degrees of freedom), and r, the radius of the cylinder. See Figure 1. The parameterizing equation is:

$$||X - P||^2 - ((X - P) \cdot A)^2 - r^2 = 0$$

with the constraints:

$$||A|| = 1$$

$$A \cdot P = 0$$

In practice, we would rather use five simple variables to parameterize the cylinder, since this parameter space is easier use. We use the polar angle, (f), and azimuth, (), of the axis direction such that:

$$A = \begin{bmatrix} \cos\phi\cos\theta \\ \sin\phi\cos\theta \\ \sin\theta \end{bmatrix}$$

We use three cases to parameterize the axis point closest to the origin, depending on which coordinate axis is closest to the cylinder axis direction. The two coordinates that do not coincide with this coordinate axis are used to parameterize the point. Without loss of generality, let us assume that $A_z > A_x$ and $A_z > A_y$. In this case, we parameterize P by P_x and P_y such that:

$$P = \begin{bmatrix} P_x \\ P_y \\ \frac{-A_x P_x - A_y P_y}{A_z} \end{bmatrix}$$

3 Searching the parameter space

We now discuss the search strategy by which primitives are located in the data. We must first specify some acceptance criterion for the primitives. One possible criterion is the number of data features that lie within some allowable error, ϵ, of the primitive. When primitives of arbitrary scale are considered, it is often useful in practice to normalize this count by the scale of the primitive (e.g. by dividing by the radius for the cases discussed above), so that there is no bias towards large primitives. We report either the best primitive according to the criterion or all primitives that surpass some threshold, t.

To find the primitives that satisfy the acceptance criterion, we consider rectilinear cells in the parameter space. It is thus imperative that bounds can be placed on parameters of the primitives that we seek, or that they are naturally bounded. This is easy to achieve if an appropriate parameterization is chosen and some simplifying assumptions are made (such as requiring that the center of the circle lie inside the image, or that the cylinder axis pass through the range data).

At the beginning of our search, the parameter space can either be considered as one large cell, or it can be divided into several smaller cells that are examined individually. (This division is necessary for cylinders, since we do not want any cells to cover more than one case in the parameterization.) Each point in the parameter space represents a possible position of a geometric primitive in the data. The parameter space cells are volumes of the parameter space and thus represent a continuous space of possible geometric primitive locations.

Each cell that is examined is tested to determine if it can contain the parameters of a primitive that satisfies the acceptance criterion. An efficient testing mechanism is used that is conservative in that it never rules out a cell that contains a good primitive, but it can fail to rule out a cell that does not contain any good primitive. This will not result in false positives since the cells are recursively examined at finer resolutions until we reach very small cells, which are tested by considering a particular primitive within the cell.

In order to test the cells efficiently, we first compute a bound on the change in the distance of any image feature from the primitive that can be caused by moving the primitive from the location specified by the center of the cell to any other position in the cell. We will denote this bound d_C for cell C. We then determine the distance, d_f, of each image feature, f, from the primitive represented by the point at the center of the cell and count how many of the features have a distance that is less than the allowable error, ϵ, plus the bound on the difference in distances described above ($d_f < d_C + \epsilon$). The cell can be pruned if this count is below the threshold set by the acceptance criterion. When the cell cannot be pruned, it is divided into two subcells by slicing it at the midpoint of one of the parameters and the subcells are considered recursively using a depth-first search.

To prevent this method from dividing the cells until they are arbitrarily small, we set a threshold on d_C. When d_C falls below the threshold, we test the cell by considering the quality of primitive at the center of the cell to see if it meets the acceptance criterion. This is equivalent to imposing some underlying discretization on the parameter space, as is done in [6, 8], and considering only those positions in the underlying discretization. It is possible that this could cause a primitive that meets the acceptance criterion to be missed, but since the cells are very small when they are tested in this manner, it is unlikely that a significant error will be made.

Note that any image feature for which it is determined that the feature cannot belong to any primitive in a parameter space cell that cannot be pruned (i.e. it does not contribute to the count as described above), need not be considered in the recursive examination of the subcells, since primitives represented by the subcells are a subset of those represented by the original cell.

3.1 Computing d_C

The computation of d_C varies depending on the class of primitives that we are trying to locate. For example, when we wish to detect circles, we use $f(X, \Gamma) = (x - x_c)^2 + (y - y_c)^2 - r^2$. Given two circles, $\Gamma_1 = [x_1, y_1, r_1]^t$

and $\Gamma_2 = [x_2, y_2, r_2]^t$, the difference in the distance of any point from these two circles can be bounded by:

$$d < \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} + |r_1 - r_2|$$

Note that this does not depend on the circles themselves, only the difference between the parameters, $\delta = \Gamma_1 - \Gamma_2$, so we can rewrite this:

$$d \leq \sqrt{\delta_x^2 + \delta_y^2} + |\delta_r|$$

Now, for any cell in the parameter space,

$$C = \{[x, y, r]^t | x_l \leq x \leq x_h, y_l \leq y \leq y_h, r_l \leq r \leq r_h\}$$

we can use this relationsl iip to place the following })01111(1 ond_C:

$$d_C \leq \sqrt{\left(\frac{x_h - x_l}{2}\right)^2 + \left(\frac{y_h - y_l}{2}\right)^2} + \left(\frac{r_h - r_l}{2}\right)$$

For cylinders, d_C is more difficult to compute, since changes in the orientation of the cylinder axis ca n change the distance of a point from the cylinder by an arbitrary amount if the point is far enough from the center of rotation. We use:

$$d_C \leq \sqrt{\left(\frac{x_h - x_l}{2}\right)^2 + \left(\frac{y_h - y_l}{2}\right)^2 + \left(\frac{z_h - z_l}{2}\right)^2} +$$

$$\left(\frac{r_h - r_l}{2}\right) + d_P \sqrt{\left(\frac{\theta_h - \theta_l}{2}\right)^2 + \left(\frac{\phi_h - \phi_l}{2}\right)^2}$$

where $(r_l, r_h), (\theta_l, \theta_h), (\phi_l, \phi_h)$, and two of (x_l, x_h), (y_l, y_h), and (z_l, z_h) are given by the boundaries of the cell. Assume, for example, that $A_z > A_x$ and $A_z > A_y$, and thus x and y are used to parameterize the point on the axis. We bound $z_h - z_l$ by examining the solutions to $P_z = \frac{A_x P_x + A_y P_y}{A_z}$ at the appropriate corners of the cell. In this equation, d_P is a bound on the distance from P (the cylinder axis point closest to the origin) to the image feature that we are comparing against. A single d_P can be used for all of the image features, or we can compute d_P for each image feature to achieve a tighter bound. Note that θ and ϕ are measured in radians.

3.2 Selecting a parameter to subdivide

We must decide, when a cell cannot be pruned, which parameter should be sliced to subdivide the cell. We should subdivide the parameter which contributes the largest amount to d_C. However, the parameters do not contribute amounts to d_C independently. We apply heuristics to estimate which of the parameters is contributing the greatest amount, and slice that parameter. For example, we prefer to split the radius before the position of the center of a circle if:

$$\sqrt{\left(\frac{x_h - x_l}{2}\right)^2 + \left(\frac{y_h - y_l}{2}\right)^2} < \sqrt{2}\left(\frac{r_h - r_l}{2}\right)$$

4 Optimizations

There are two primary methods by which we improve the speed of the parameter space search. The first is to consider a hierarchy of the image features that allows many of the features to be pruned quickly for a particular parameter space cell and the second is the use of random sampling to reduce the overall amount (If the data that is processed by the full algorithm.

4.1 Image feature hierarchy

It is possible to rule out a cell of image features for a particular parameter space cell by examining a single image location using a method similar to the one by which cells in the parameter space are pruned after examining only a single position in the cell. Consider a cell, D, in the image feature space. Let c_D be the center of this cell, d_D be the distance from c_D to the furthest image feature in the cell, and d_{c_D} be the distance from c_D to the primitive represented by the center of a parameter space cell, C. Define d_C as above, except that this bound now must apply to the image features in D. If $d_{c_D} > d_D + d_C + e$, then no image feature in the image cell can match any primitive in the parameter space cell up to the error, e, and we can thus prune the image cell for this parameter space cell and any of its subcells that are examined.

To take advantage of this idea, we build a hierarchy of image feature cells. See Figure 2. This is a binary tree, where each node corresponds to a cell in the image feature space. The root of the tree is a cell just large enough to contain all of the image features. Each cell that contains more than one image feature has two children that are roughly half its size. Note though, that the union of the subcells need not be equivalent to the parent cell, since we need only ensure that the subcells cover all of the image features in the original cell. Each individual feature is a leaf of the tree. We build this tree recursively by subdividing the cells at the midpoint of the longest axial direction. Each subcell is then contracted such that it is just large enough to contain all of the data features within it.

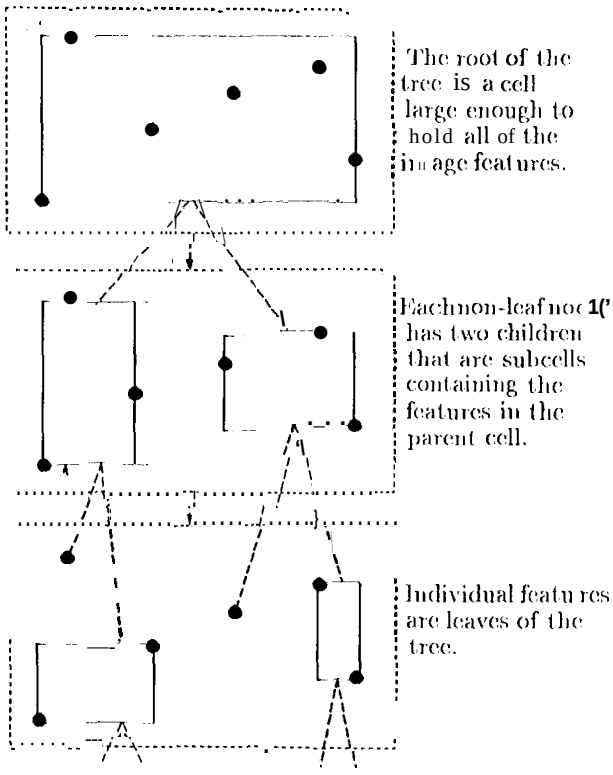


Figure 2: The image features are recursively subdivided to form a hierarchy similar to the parameter space ((,11S.

Now, when a particular parameter space cell is examined, we search this tree of cells in the image space, rather than considering each feature individually. This is performed in a similar manner to the search over the parameter space cells. For each image space cell that is examined, we determine if it can be pruned. If the cell can not be pruned, then we examine its subcells, unless it is a leaf, in which case we increment the count on the number of features that cannot be pruned.

4.2 Random sampling

In many cases, there is more data than necessary to extract a geometric primitive. For example, a cylinder that covers only a 25x100 patch in a range image yields 2500 points on the cylinder, which is far more than we need to extract the cylinder. We can use random sampling techniques to reduce the amount of data that is examined, while only slightly decreasing the accuracy of the techniques. To accomplish this, we sample some set of the image features at random, which are processed as described above. If we find a cylinder that matches enough of the data features then we test the cylinder using all of the data features.

We assume that only geometric primitives consisting of at least some fraction, α, of the image features are salient and must be located. If we sample s features and test the full cylinder if γs of the sampled features belong to a primitive then the probability that a primitive is missed due to sampling is:

$$\sum_{k=0}^{\lfloor \gamma s \rfloor - 1} \binom{s}{k} \alpha^k (1-\alpha)^{n-k} \leq e^{-(1-\frac{\gamma}{\alpha})^2 s\alpha/2}$$

The above bound is a direct result of Proposition 2.4(a) in [1], which follows from Chernoff's bound. This provides us with a means of determining what value should be used for γ given a maximum error rate ϵ. In practice, we compute the γ that satisfies this error rate numerically, since a tight bound is desirable. For example, a 256x256 range image containing a cylinder comprised of 2500 pixels has $\alpha = 0.0381$. If 1000 points are sampled and a probability of failure no more than 0.01 is desired, then $\gamma \leq 0.0260$ can be used to achieve this.

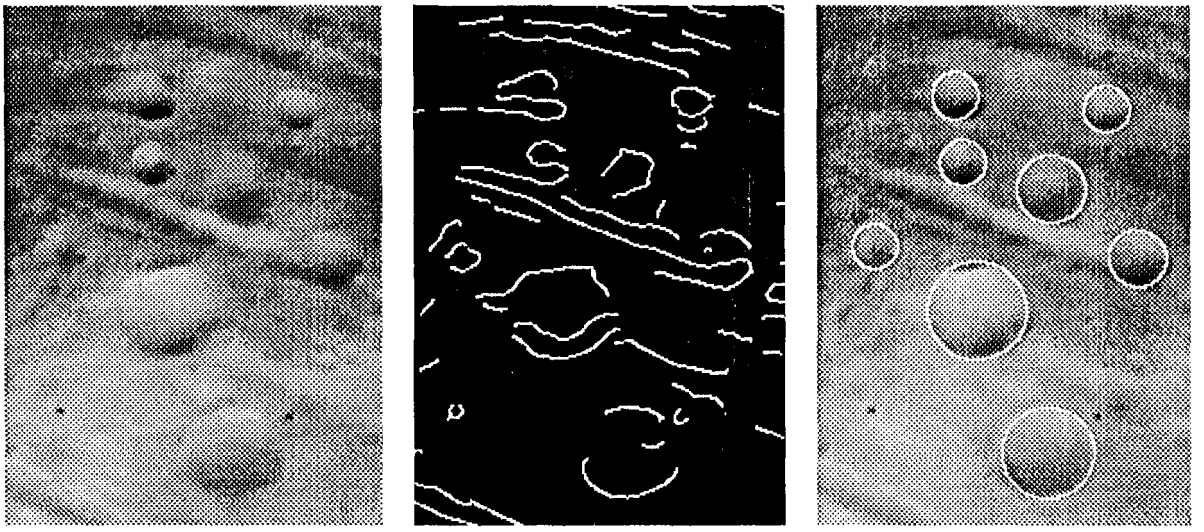
5 Results

We have applied these techniques to two problems. First, we have examined the extraction of craters on planetary bodies through the detection of circles in two dimensional edge data and, second, we have examined finding surface-lying ordnance by extracting cylinders from three-dimensional range data.

5.1 Detection of craters

It is desirable to be able to detect craters on planetary bodies in several applications. Some examples include mapping of planetary bodies, geological studies, and optical navigation of spacecraft. One approach to accomplishing this is to perform edge detection on an image of the surface of a planetary body and detect the circles that are present in the edge image. Note that this assumes that the optical axis is close to perpendicular to the craters, so that, the craters appear as circles rather than ellipses. When this is not the case, similar techniques can be used to extract ellipses from edge data.

In this application it is useful to incorporate knowledge about the lighting direction in the extraction process, since the gradient orientations at the edges of craters will have roughly the same orientation as the lighting direction (positive dot-product), while shadow edges will have the opposite orientation (negative dot-product). We can thus screen out the shadow



(a) (b) ((')

Figure 3: Finding craters by locating circles in edge images. (a) An image of the surface of Phobos (the larger of the two moons of Mars). (b) The edges detected in the image. (c) The craters detected overlaid 011 the original image.

edges through the usc of this information. Even if the lighting direction is not known explicitly, its direction with respect to the image plane can be estimated by examining a weighted average of the gradient directions in the image.

Figure 3 shows an application of these techniques to detecting craters on the surface of Phobos (a 1110011 of Mars). Edges were detected in an image from *Viking Orbiter* using a Canny-like edge detector [5]. We searched for the circles t]] at had $\frac{f'}{\tau} \geq 1$, where f' is the number of image pixels that matched the circle up to a maximum error of 1.5 pixels and r is the radius of the circle. We did not usc the random sampling techniques for this case, since the volu me of the data was not large. Eight circles were found that met the acceptance criterion. The extraction techniques required approximately 4 seconds 011 a SPARCstation™20 for **this case.**

As can be seen, these techniques yield good performance in detecting relatively recent craters that have significant edge structure. Older craters may be problematic for the current techniques when the edge structure has deteriorated. We may be able to perform recognition in this case by treating the grey-level data as a three-dimensional surface and detecting the craters based on the shading information, if the angle of incidence of the lighting is known. This technique is similar to a template matching approach, but it would handle model error better and yield potentially faster performance.

5.2 Finding surface-lying ordnance

A second application that we have examined is locating unexploded, surface-lying ordnance using passive stereo imagery for the purpose of semi-autonomous remediation. We use a binocular stereo system designed for planetary rovers [12] to generate range images of outdoor scenes. Cylinders are extracted from the range data to help determine if there is an unexploded bomb present in the image. 111 this case, we assume that the radius of the bomb and the orientation of the ground plane are known, so that the search space is restricted to three degrees of freedom. We also use the known geometry of cylinders to prevent portions of the cylinder that should be self-()(clu(i('(1 from matching points in the image.

Figure 4 shows an example image containing a bomb where we have applied the cylinder detection techniques. In this case, we sampled 5% of the range data and used numerical techniques to determine an appropriate **till'('sllol(l for** testing **t 0** the entire data set. Approximately 7 seconds were required to search the parameter space (in a SPARCstation™20. A cylinder was detected at the location of the bomb. No other cylinders were detected.

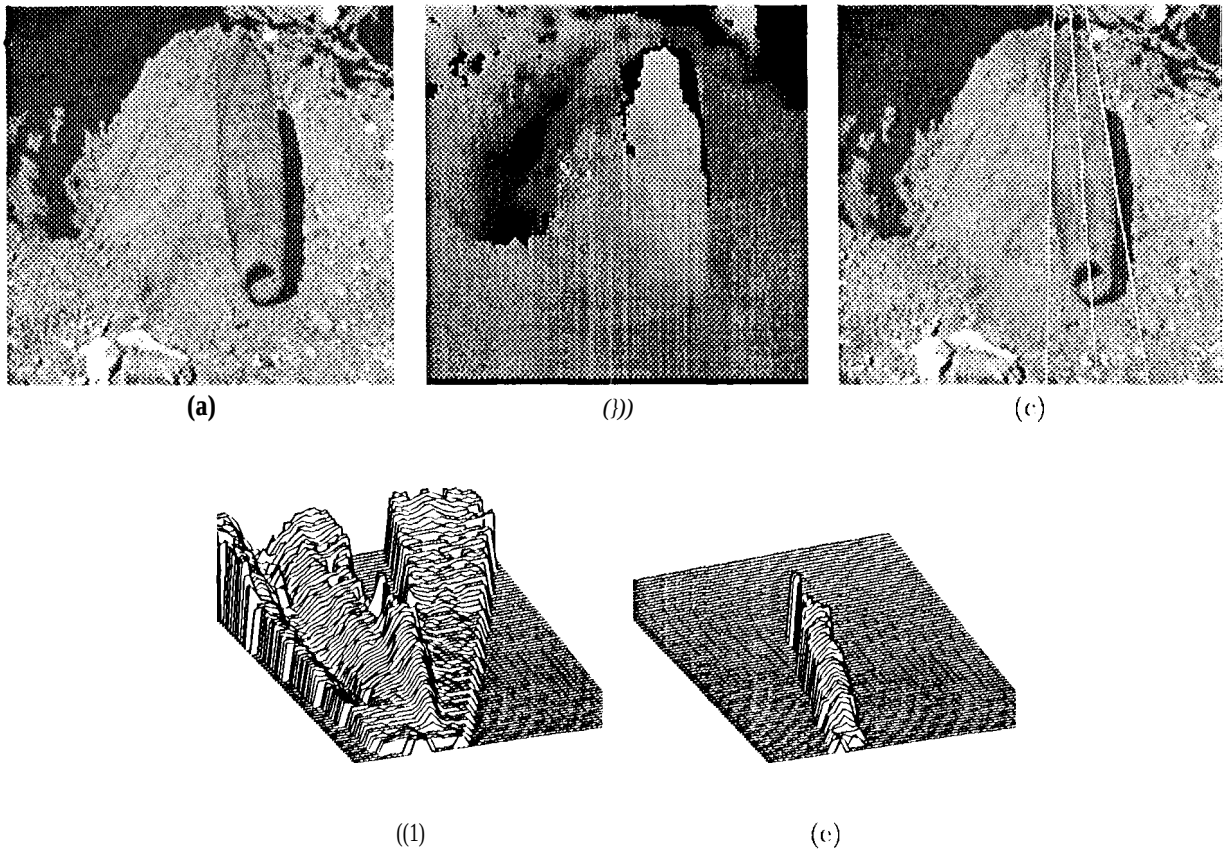


Figure 4: The use of cylinder detection to find unexploded ordnance. (a) The left image of a stereo pair containing surface-lying ordnance. (b) The height of each pixel is determined using binocular stereo. White pixels are locations where no range information was found. (c) A cylinder meeting the acceptance criterion was found at the location of the bomb. This image shows the axis and boundaries of the cylinder. (d) Plot of the range points computed in the image. (e) Plot of the cylinder points detected in the image.

6 Summary

This paper has explored the detection of geometric primitives in image data. We have adopted a search strategy that has been used recently in object recognition [4, 8], where the parameter space is recursively divided and pruned. Whereas this strategy was previously applied to detecting discrete sets of points in image data, here we have applied it to locating primitives that are represented by parameterized manifolds in the image data. This allows geometric primitives to be robustly and efficiently extracted from noisy data with large amounts of distracting data, without requiring an initial estimate of the primitive locations, where no previous algorithm combined these qualities. Further improvements are gained through the use of a hierarchical representation of the image features that allows them to be efficiently processed and the use of random sampling to reduce the volume of data that must be processed. These techniques have been applied to the detection of craters on planetary bodies by extracting circles from two-dimensional edge data and locating surface-lying ordnance by extracting cylinders from full, three-dimensional range data.

Acknowledgments

The authors thank Ed Brown for supplying an inert bomb for our use. We also appreciate Todd Litwin's efforts in developing the camera calibration and binocular stereo code that was used in generating range maps.

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the Construction Automation and Robotics Branch of Wright Labora-

tory at Tyndall Air Force Base, Panama City, Florida, through an agreement with the National Aeronautics and Space Administration.

Reference herein to any specific commercial product, process, or service by trade name, trademark manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government, or the Jet Propulsion Laboratory, California Institute of Technology.

References

[1] D. Angluin and L. G. Valiant. Fast proba balistic algorithms for Hamiltonian circuits and matchings. *Journal of Computer and System Sciences*, 18:155193, 1979.

[2] P. J. Besl and R. C. Jain. Segmentation through variable-order surface fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(2):167-192, March 1988.

[3] R. C. Bolles and M. S. Fischler. A RANSAC-based approach to model fitting and its application to finding cylinders in range data. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 637 642, 1981.

[4] T. M. Breuel. Fast recognition using adaptive subdivisions of transformation space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 445 451, 1992.

[5] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679 697, November 1986.

[6] 'J'. A. Cass. A robust implementation of 2d model-based recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 879 884, 1988.

[7] D. S. Chen. A data-driven intermediate level feature extraction algorithm. *IEEE Transactions 071 Pattern Analysis 07/d Machine Intelligence*, 11(7):749 758, July 1989.

[8] D. P. Huttenlocher and W. J. Rucklidge. A multi-resolution technique for comparing images using the Hausdorff distance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 705 706, 1993.

[9] J. Illingworth and J. Kittler. A survey of the Hough transform *Computer Vision, Graphics, and Image Processing*, 44:87 *116*, 1988.

[10] V. F. Leavers. Which Hough transform? *CVGIP: Image Understanding*, 58(2):250 *264*, September 1993.

[11] H. Li, M. A. Lavin, and R. J. Le Master. Fast Hough transform: A hierarchical approach. *Computer Vision, Graphics, and Image Processing*, 36:139 161, 1986.

[12] L. Matthies. Stereo vision for planetary rovers: Stochastic modeling to near real-tilfj, implementation. *International Journal of Computer Vision*, 8(1):71 91, July 1992.

[13] P. Meer, D. Mintz, A. Rosenfeld, and D. Y. Kim. Robust regression methods for computer vision: A review. *International Journal of Computer Vision*, 6(1):59 70, 1991.

[14] G. Roth and M. D. Levine. Extracting geometric primitives. *CVGIP: Image Understanding*, 58(1):1 22, July 1993.

[15] F. Solina and R. Bajcsy. Recovery of parametric models from range images: The case for superquadrics with global deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2):131 *147*, February 1990.